## AMENDMENTS TO THE CLAIMS

Please amend the claims as follows.

1. (Currently Amended) A method [[for]] to trac[[ing]]e an instrumented program, comprising:

triggering a[[n]] trap instruction in the instrumented program during tracing of the instrumented program;

transferring control of the instrumented program to a trap handler associated with the trap instruction;

calling into a tracing framework, by the trap handler, to perform a tracing operation associated with the trap instruction;

performing the tracing operation to obtain tracing information, wherein the tracing information is used to analyze the instrumented program; and

emulating, after performing the tracing operation, an original instruction in the instrumented program using ~~corresponding to the trap instruction in~~ the trap handler,

wherein the original instruction is associated with the trap instruction, and

wherein the original instruction relates to creating or dismantling a stack frame.

2. (Currently Amended) The method of claim 1, further comprising:

replacing the original instruction with the trap instruction in the instrumented program.

3. (Cancelled)

4. (Currently Amended) The method of claim 1, wherein emulating the original instruction comprises emulating a pushl instruction.

5. (Currently Amended) The method of claim 4, wherein emulating a pushl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a location in the stack frame;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into the stack pointer location;

loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer; and

loading a base pointer into two locations after the stack pointer location.

6. (Original) The method of claim 5, further comprising:

decrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

7. (Original) The method of claim 5, wherein the instruction pointer is loaded at the stack pointer location.

8. (Currently Amended) The method of claim 1, wherein emulating the original instruction comprises emulating a enter instruction.

9. (Currently Amended) The method of claim 8, wherein emulating an enter instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location is corresponds to a location in the stack frame;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into the stack pointer location;

loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer;

loading a base pointer into two locations after the stack pointer location; and

loading the base pointer into a base pointer register.

10. (Original) The method of claim 9, further comprising:

decrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

11. (Original) The method of claim 9, wherein the instruction pointer is loaded at the stack pointer location.

12. (Currently Amended) The method of claim 1, wherein emulating the original instruction comprises emulating a popl instruction.

13. (Currently Amended) The method of claim 12, wherein emulating a popl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a location in the stack frame;

loading a base pointer obtained from three locations after the stack pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into three locations after the stack pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into two locations after the stack pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer; and

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location.

14. (Original) The method of claim 13, further comprising:

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

15. (Original) The method of claim 13, wherein the instruction pointer is loaded at the stack pointer location.

16. (Currently Amended) The method of claim 1, wherein emulating the original instruction comprises emulating a leave instruction.

17. (Currently Amended) The method of claim 16, wherein emulating a leave instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a first location in the stack frame;

obtaining a base pointer location, wherein the base pointer location corresponds to a second location in the stack frame;

loading a base pointer obtained at the base pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into the base pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into one location before the base pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at two locations before the base pointer location; and

loading the instruction pointer at three locations before the base pointer.

18. (Original) The method of claim 17, further comprising:

setting the stack pointer location to three locations before the base pointer location;

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

19. (Currently Amended) A system [[for]] to trac[[ing]]e an instrumented program, comprising:

the instrumented program comprising at least one trap instruction associated with an original instruction, wherein the original instruction relates to creating or dismantling a stack frame;

a thread configured to execute the instrumented program;

[[and]]

a trap handler configured to halt execution of the thread when the trap instruction is encountered during tracing of the instrumented program, call into a

tracing framework to perform a tracing operation associated with the trap instruction, and to emulate the original instruction associated with the trap instruction; and

the tracing framework configured to perform the tracing operation to obtain tracing information, wherein the tracing information is used to analyze the instrumented program.

20. (Cancelled)

21. (Currently Amended) The system of claim 19, wherein the original instruction is replaced with [[a]] the trap instruction in the instrumented program.

22. (Currently Amended) The system of claim 19, wherein emulating the original instruction comprises emulating a pushl instruction.

23. (Currently Amended) The system of claim 22, wherein emulating a pushl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location is corresponds to a location in the stack frame;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into the stack pointer location;

loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer; and

loading a base pointer into two locations after the stack pointer location.

24. (Original) The system of claim 23, further comprising:

decrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

25. (Original) The system of claim 23, wherein the instruction pointer is loaded at the stack pointer location.

26. (Currently Amended) The system of claim 19, wherein emulating the original instruction comprises emulating a enter instruction.

27. (Currently Amended) The system of claim 26, wherein emulating an enter instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location is corresponds to a location in the stack frame;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into the stack pointer location;

loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer;

loading a base pointer into two locations after the stack pointer location; and

loading the base pointer into a base pointer register.

28. (Original) The system of claim 27, further comprising:

decrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

29. (Original) The system of claim 27, wherein the instruction pointer is loaded at the stack pointer location.

30. (Currently Amended) The system of claim 19, wherein emulating the original instruction comprises emulating a popl instruction.

31. (Currently Amended) The system of claim 30, wherein emulating a popl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a location in the stack frame;

loading a base pointer obtained from three locations after the stack pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into three locations after the stack pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into two locations after the stack pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer; and

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location.

32. (Original) The system of claim 31, further comprising:

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

33. (Original) The system of claim 31, wherein the instruction pointer is loaded at the stack pointer location.

34. (Currently Amended) The system of claim 19, wherein emulating the original instruction comprises emulating a leave instruction.

35. (Currently Amended) The system of claim 34, wherein emulating a leave instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a first location in the stack frame;

obtaining a base pointer location, wherein the base pointer location corresponds to a second location in the stack frame;

loading a base pointer obtained at the base pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into the base pointer location;

loading a code segment (CS) value stored one location after the stack pointer location into one location before the base pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at two locations before the base pointer location; and

loading the instruction pointer at three locations before the base pointer.

36. (Original) The system of claim 35, further comprising:

setting the stack pointer location to three locations before the base pointer location;

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.